THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Testing and Integration

By H. A. HILSINGER, K. D. MOZINGO, C. F. STARNES, G. A. VAN DINE

(Manuscript received July 16, 1976)

This article describes the general approach that was taken in testing and integrating of the 1A Processor. Included is a survey of the computer aids and test facilities that were used in performing the testing and integration tasks.

I. INTRODUCTION

Planning for system integration and testing started at the beginning of the 1A Processor[1] development. Much of this planning was based on experience gained in the development of No. 1 ESS[2] and other stored program systems.[3,4] This experience indicated that the availability of high-quality tests, test facilities, and comprehensive test plans are critical to the orderly development of a high-quality system. In the development of the 1A Processor, tests and test facilities were designed concurrently with the design of the processor hardware and software. The objectives were timely integration of hardware and software components into a working system and thorough verification of the hardware and software designs.

The general approach to integration of the processor system was to design and test in parallel the various system components and to integrate incrementally these components into a working system. This paper describes the integration process in terms of three levels, namely:

(*i*) Circuit-pack verification and testing.
(*ii*) Frame verification and testing.
(*iii*) System integration and testing.

At each level of integration, computer aids played a major role in design verification before physical prototypes were built. Using computer aids, errors were uncovered in early stages of the design when corrections could be made with least impact on development resources and schedules. Substantial physical testing was done at each level of integration to verify proper implementation of the design not only in a normal environment but at various design limits. Special change procedures were developed for each level of integration to provide quick temporary fixes as well as permanent corrections for design errors.

Early in the project, it was recognized that many of the requirements for prototype testing at Bell Laboratories were very similar to requirements for new frame and installation testing to be done by Western Electric manufacturing and installation groups. Therefore, many of the tests and test facilities were developed jointly by Bell Laboratories and Western Electric engineers as common multiapplication designs. This approach was considerably different from that used to develop other systems, particularly with respect to processor diagnostic test design.

The 1A Processor diagnostic tests were designed jointly by Bell Laboratories and Western Electric as a common diagnostic data base for use in factory frame testing and installation (XRAY) testing, as well as for incorporation in the processor software for maintenance of an operating office. With this approach, a high-quality diagnostic data base was produced early in the development with much less manpower than would have been required to develop separate designs for the three applications.

This paper describes the Bell Laboratories application of the common tests and test facility designs to prototype testing and design verification. Not within the scope of this paper are the Western Electric applications, the numerous exploratory tests conducted prior to developing the 1A Processor, or the No. 4 ESS[5] and No. 1A ESS[6] system tests used to verify overall system functions in the various applications of the 1A Processor.

II. CIRCUIT-PACK VERIFICATION AND TESTING

2.1 Computer aids

2.1.1 Machine-aided design system

The circuit density and logic complexity achieved with 1A technology[7] has resulted in a need for computer aids at virtually every step of the design and manufacture. At the heart of the circuit-pack verification and testing process is a digital-circuit-pack machine-aided-design system (MADES). MADES generates design-verification information, docu-

mentation, manufacturing artmasters, and test data from a common data base built from the engineer's logic description of the circuit.

Figure 1 illustrates the organization and functions of MADES. The logic description of a circuit is input to the logic analysis program (LAMP)[8] and a simulation is performed. When the engineer is satisfied that the LAMP simulation has verified the design intent, an automatic placement program uses the LAMP logical description to:

(*i*) Partition gates to silicon-integrated-circuit (SIC) chips.
(*ii*) Place SIC chips on the ceramics.
(*iii*) Assign load resistors and other circuit components.
(*iv*) Place schematic symbols.
(*v*) Build the design data base.

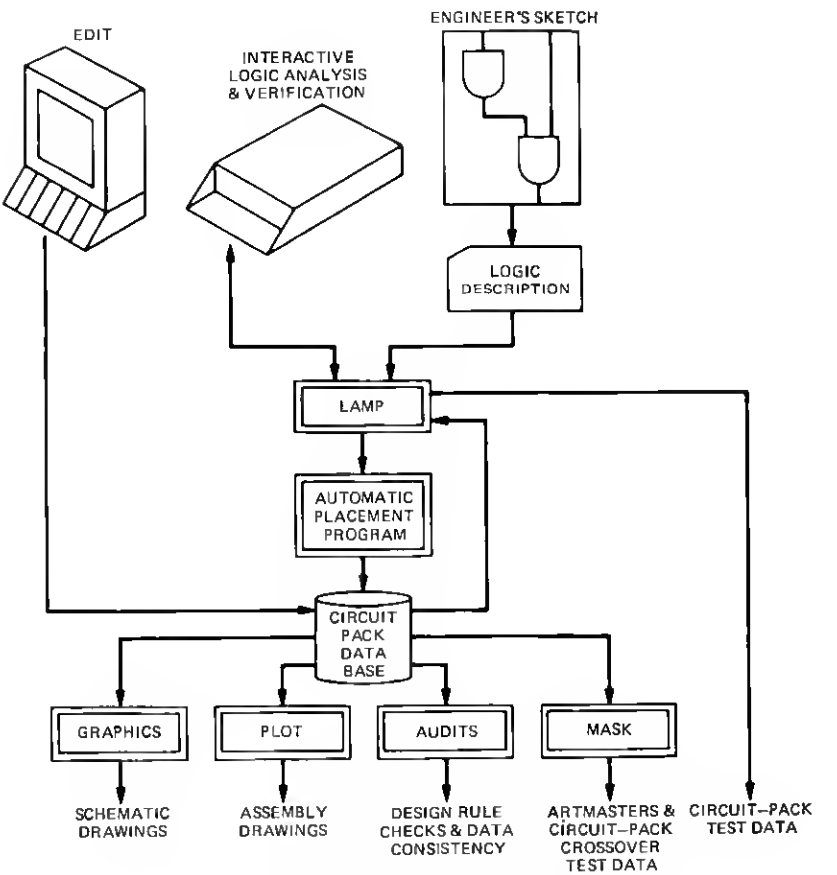

Fig. 1—Digital-circuit-pack, machine-aided design system.

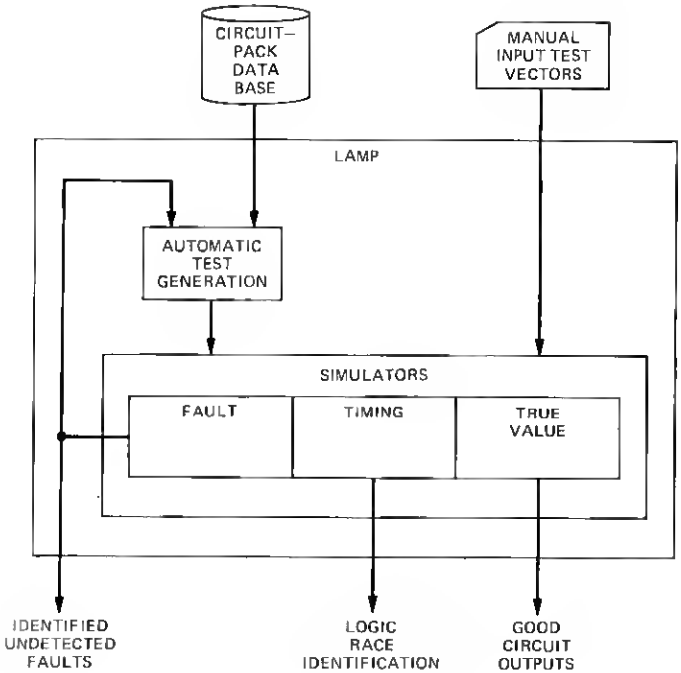

Fig. 2—Circuit logic-analysis program.

From the design data base, a variety of computer programs can be executed to produce the final art work, documentation, and test data used in manufacturing. This highly automated process also provides for manual intervention for handling nonstandard cases. Audits are provided to assure that design rules are not inadvertently violated and that the data base can pass a sanity check for consistency.

2.1.2 Circuit-pack test generation

The LAMP program is used both as a general input interface to MADES and as a logic simulator used to verify design intent with the output results constituting circuit-pack test data. Figure 2 illustrates in greater detail the design verification capability of LAMP. The engineer first manually designs and simulates on the true value (no-fault) simulator a set of test vectors for logic-verification purposes. Any logic errors uncovered are corrected before proceeding further. Additional test vectors for fault detection are then generated either manually or by using the automatic test generation (ATG)[9] facility in LAMP. All test vectors are then applied to two fault simulators. The classical fault simulator systematically induces gate stuck-high and stuck-low faults. The shorted

fault simulator induces adjacent-lead and crossover-short faults derived from circuit-pack artmaster information. If the test vectors are not adequate to detect all of the simulated faults, the undetected faults are identified and fed back to ATG or the engineer as appropriate. In some cases, the engineer incorporates logic changes to improve testability. This process is iterated until all or a high percentage of the faults are detected by the test vectors. The timing simulator is then used to detect any input race conditions, which are subsequently eliminated, and the true-value simulator is run to provide the "output" data for use on the circuit-pack test facility (described in 2.2.2 and 2.2.3).

This logic analysis program is capable of bandling highly sequential as well as combinational logic and has been proven to be cost effective both in logic-design and test-design verification and for generating manufacturing data.

2.2 Circuit-pack testing

Circuit-pack test facilities were developed for use at Bell Laboratories as well as for use at the Western Electric factory. At Bell Laboratories these facilities were primarily used for testing and verification of "quick-fix" circuit-pack modifications to facilitate rapid turnaround of hardware design changes.

2.2.1 Crossover testing

The first level of pack testing is an electrical continuity check of the crossovers on the metallized ceramic. This is done before component bonding using a numeric controlled probe table driven from data generated by MADES.

2.2.2 Pass/fail circuit-pack test facilities

A circuit-pack test facility was developed for Bell Laboratories and Western Electric use in test and repair of logic packs. The basic test procedure is to apply the sequence of LAMP-generated tests to the circuit-pack terminals under computer control and compare the actual response with the expected good-pack response. The development model was also equipped with a high-performance reed matrix that allowed automatic connection to an auxiliary programmable instrument trailer for precision timing and parameter testing.

Figure 3 is an illustration of the physical prototype hardware used during the development phase of the 1A Processor. The fan-shape physical design of the high-performance reed matrix dramatically reduces the backplane wire lengths over conventional in-line pluggable connector arrangements. This was essential to control parasitic effects and get good impedance matching through the matrix. A programmable

Fig. 3—Computer-controlled, circuit-pack test and diagnostic system.

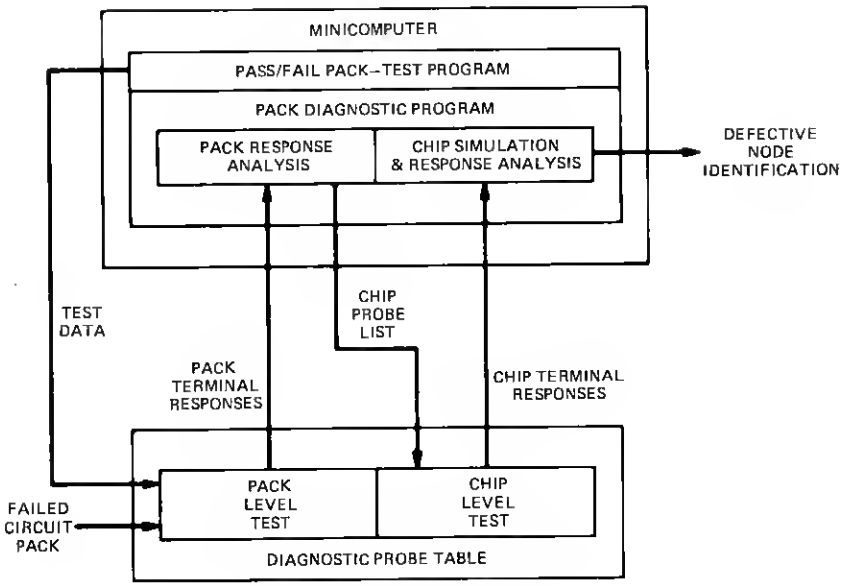

Fig. 4—Circuit-pack diagnostic system.

instrument trailer is equipped with a commercially available pulse generator, sampling oscilloscope, strobing voltmeter, automatic RLC bridge, frequency synthesizer, and programmable counter. The circuit-pack test facility provides the electronic interface for the logic tests and the ability to switch analog test equipment. The test system is capable of verifying the hardware design and separating good product from bad.

2.2.3 Fault diagnosis and isolation

To assure that this new technology could be manufactured economically, a means of effectively isolating and repairing faults was necessary. Conventional manual trouble shooting is difficult because of the complex and highly sequential nature of the logic and because of the microscopic dimensions of the physical components. To solve the physical probing problem, a computer-controlled probe table was developed capable of locating and contacting the beam-lead terminals on an individual silicon-integrated-circuit chip on the pack (Fig. 4). The defect isolation is accomplished in three progressive steps of testing starting at the pack level, progressing to the chip level, and concluding with some analog tests at the defective node.

The input to the diagnostic software system consists of basically a logic description, terminal and chip assignment information, and LAMP-generated circuit-pack tests. At the pack-test level, defect-analysis algorithms establish a list of potentially faulty chips based on failing responses at the pack terminals. This list of suspect chips, with a most-probable-faulty ordering, determines the probing sequence for chip-level testing.

Chip-level testing selects the next chip site to be probed, positions the probe (under program control), and monitors the chip terminals while the same (pack-level) test vectors are applied at the pack terminals. The chip inputs are measured and sent to a real-time logic simulator in the minicomputer, which then determines what the correct logic response should be. The result is then compared with the measured chip outputs. This procedure continues until there is a mismatch between a chip's response and the simulator. The bad node is then identified by chip location, beam-lead terminal number, failed-test number, and failed state (high or low). Note that the resolution is to the node only. A further analysis must be made to determine which chip or ceramic metallization detail tied to the node is at fault.

It should be emphasized that it is this second-level diagnosis (which isolates the bad node) that is by far the most important. The first level reduces the amount of probing but could be omitted. The second level eliminates hours of laborious tracing through a complex sequential circuit to find a defect that may not propagate a wrong response to pack terminals for over 100 tests.

One reason why this system is both unique and cost effective is that the simulation is done in real time on the minicomputer and so does not require stored data from previous exhaustive simulation at the pack level. This technique results in a tremendous reduction of the amount of data that must be maintained for each pack code. It is based on the philosophy that it is easier to generate, as needed, a small segment of simulation in real-time than to store and search results of an exhaustive simulation previously done on the entire circuit pack. This approach is also very effective in a multiple-defect environment. Because the chip simulator uses measured input values, it does not depend on correct or assumed input values to generate the expected true-value logic response. This allows testing to proceed beyond the first defect, which can result in locating additional faults deeper in a logic chain.

The third level of defect resolution is a series of analog tests designed to determine which device associated with a node is faulty. The technique used is to make sensitive voltage measurements through Kelvin probes to determine the current flow and device influence at a node. This level of testing could be automated, but since it is so easily accomplished by the test-facility operator, it is typically left as a manual operation.

III. FRAME VERIFICATION AND TESTING

The next level of assembly is a functional unit made up of a collection of circuit packs interconnected at the backplane and mounted on a structural frame. Some frames contain only one functional unit, whereas others contain more than one unit. The term "frame verification and testing" is used somewhat loosely in this paper, since "unit verification and testing" is implied for multiunit frames.

3.1 Computer aids for frame-design verification

3.1.1 Frame-design data base

Basic design information for each 1A Processor frame was assembled into a frame-design data base on the Bell Laboratories computer system. This data base then served as the primary source for automatic generation of data needed for design verification and manufacture.

Design verification prior to the assembly of prototype frames was accomplished through logic simulation (described in 3.1.2) and through the designer's inspection of automatically produced frame drawings. After automatic artmaster and wire layout generation, audit programs were run to obtain wire length and adjacent wire exposure measurements for evaluation of noise and crosstalk margins. A timing-analysis program was also used to evaluate worst-case timing margins on critical logic chains, taking into account such parameters as wire lengths, gate fan-out/loading, and worst-case gate delays.

3.1.2 Frame logic simulation

Logic simulation at the frame level played an important role in verification of initial logic and diagnostic test designs. Many circuit- and diagnostic-design errors were uncovered and corrected before the physical frame prototypes were constructed. Figure 5 shows the major components of frame-level simulation using LAMP on an IBM 360/370 TSS.

Primary inputs to LAMP were the logical model of the frame or unit to be simulated and the input vectors to he simulated. The logic model was assembled by the logic simulation language (LSL) compiler in LAMP, primarily from logic interconnection data extracted from the frame-design data base. Some errors in the data base were uncovered by consistency checks included in the LSL compiler. Input vectors were assembled for simulation from the macro language test statements in the diagnostic data base using the macro language facilities of the switching assembly program (SWAP).

Several runs were made on the LAMP simulation system. Initial runs used the true-value (no-fault) simulator and resulting output vectors

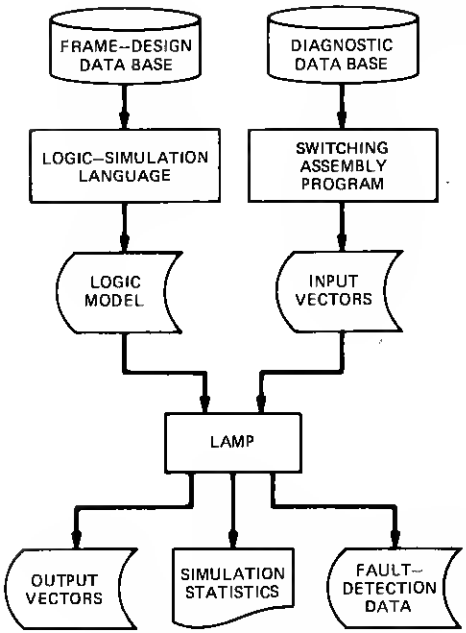

Fig. 5—Frame-logic simulation.

were compared with expected results (also in the diagnostic data base) for the diagnostic tests simulated. When discrepancies were found, the logic and diagnostic designers worked together to resolve them. Initially, only a few sample tests were simulated and many discrepancies found were due to weaknesses of the logic model. These required manual editing to correct. For example, digital models for certain analog circuits had to be added and "tuned" to properly simulate the analog circuits. After model weaknesses were corrected, more diagnostic tests were run. A number of discrepancies between simulation output vectors and expected results were traced to frame-design and diagnostic-design errors. Such errors were temporarily corrected by editing the logic model and input vectors in order to proceed with further simulation. Permanent corrections to the frame-design and diagnostic data bases could thus be made at a later date without delaying simulation.

After true-value verification of the logic model and the diagnostic test vectors, a second set of runs was made on the LAMP logic simulator to evaluate diagnostic fault-detection capabilities. As a result of these fault-simulation runs, additional tests were added to the diagnostic data base and some logic changes were applied early in the design to enhance fault detection.

3.2 Physical frame testing

As previously indicated, the processor diagnostic tests were designed from the outset to be general-purpose in application. The tests are comprised of sequences of input bit combinations (words) applied to a 1A Processor frame via its external send bus connectors. The test sequences are applied to the frame via these connectors by the central control (CC) in an operating system environment. In the frame-test environment the frame-test facility ties into these bus connectors and applies the same test sequences.

Associated with each test is an expected-result word which a fault-free frame should return on its reply bus. In the frame-test environment, the frame-test facility ties into the frame reply-bus connectors and checks test responses from the frame in the same manner that the CC does in an operating system.

In the case of the CC frame itself, additional access is available in the form of connectors which, in the office environment, connect to the other CC. This port provides the necessary communications by which the CCs compare results and test each other in case of disagreement.

As with the initial diagnostic tests, the frame-test facilities were designed by Bell Laboratories and Western Electric personnel working together. The resulting test set designs were applied both to early frame testing at Bell Laboratories and to frame testing at the Western Electric factory.

The frame-test facility consists of an interface unit and a minicomputer whose disk and core memories hold all of the test inputs and expected-results data. This combination is designed to emulate the operation of the active CC in applying a test and checking for the correct response with bus timing equivalent to that in an operating system. A printer associated with the minicomputer outputs the test results. The very large number of tests required to check out units as complex as 1A Processor frames virtually precludes a purely manual test procedure. Nevertheless, a manual control and a display panel are provided on the interface unit to allow manual intervention when necessary to isolate especially difficult problems.

In operation, diagnostic tests are run from the minicomputer and the output test results are printed. Any trouble area is quickly isolated because the test sequences are organized into test phases, each of which checks out a very specific area (or function) of frame logic. Once a repair is made to a failed area, the failing test phases are rerun. The procedure works well, even in the case of multiple faults, because the diagnostic test organization generally administers tests to an area of logic only through circuitry that has passed previous testing.

Figure 6 shows the arrangement of a typical frame-test facility.

Fig. 6—Frame-test facility.

IV. SYSTEM INTEGRATION AND TESTING

Substantial testing/verification at the system level was not done until the first prototype processor was assembled, although some system-level functions were partially verified by simulation. For example, instruction fetch and execution were simulated on LAMP using the central control logic model connected to a functional model of a program store. Some special-purpose simulation programs were also written. For example, a FORTRAN program was written to simulate file store read/write transfers and direct memory-access interface with central control. This simulation was used in selecting an optimum algorithm for allocating call store and program store cycles to central control and file store.

Integration and testing of the 1A Processor hardware[10,11] and software[12,13] was done in three phases: initial hardware and diagnostic testing, software integration, and system test and evaluation.

4.1 Initial hardware and diagnostic integration

The objective of initial testing on the first processor assembled was to verify hardware interconnections, basic instruction execution, and diagnostic test capability. In planning for this initial integration phase, it was recognized that a special software "operating system" with simple input/output and diagnostic control capabilities was needed to bring up the initial processor. It was also recognized that essentially the same operating system was needed for Western Electric factory and installation testing. To meet these common needs, a software package called the installation test system (ITS) was developed by a team of Bell Laboratories and Western Electric engineers.

A primary objective of the ITS control software design was to provide tools for incremental buildup and testing from a minimum processor configuration to a full system configuration, including application peripheral units. The ITS package makes it possible to get the basic operating system cycling on a minimum configuration. This consists of one central control, one program store, one call store, and a special IO terminal interface board inserted in the central control. The terminal interface board provides a simple IO capability via direct access to the internal buffer bus of the central control. Special utilities are provided for use in pumping of program store from cassette tape or remote computer via the special terminal interface until the 1A Processor's system reinitialization (SR) function is operational. Other utilities are provided for bringing on-line additional program stores, call stores, and disks and pumping them from disk or tape.

As mentioned previously, the diagnostic test tables assembled from the multiapplication diagnostic data base provide the primary vehicle for factory system testing and installation testing. A number of options

are provided by the ITS control software for interactive execution of the diagnostic tests. The diagnostic test tables can be paged from either disk, magnetic tape, or additional program store(s). Input commands are provided for halting diagnostic execution at a specified location, dumping internal registers or memory, and looping on a specified segment of tests with an oscilloscope synchronization pulse generated on a specified test.

ITS also provides for simultaneously running tests, including diagnostic looping, on several units in a time-shared multitask mode. With this multitask capability, a crew of testers can bring up several new frames in parallel once the minimum processor capability has been established.

Although the ITS control software was designed primarily for Western Electric factory and installation testing, it was an ideal operating system for debugging the first prototype processor and the diagnostic tests. The interactive multitask features were especially valuable for hands-on parallel debugging of subtle bardware or diagnostic problems that in general could not be resolved using a "batch run" mode of operation.

The initial hardware and diagnostic integration was completed and a tested processor with debugged diagnostic tests was delivered to the No. 4 ESS system-integration group within one year of receipt of the first processor frames from the factory.

4.2 Processor software integration

It is not within the scope of this paper to describe such computer aids as the editor, assembler/compiler, loader, etc. that were used in the design of the 1A Processor software (common to all applications). It is, however, appropriate to mention the use of the 1A Processor simulator.

A prerequisite to the introduction of program code in the system lab was that it first be unit tested on the 1A Program simulator. This simulator was implemented on an IBM 360/370 TSS and provided functional-level simulation of most 1A Processor instructions and associated register and memory operations. Using this simulator, much of the program unit testing overlapped initial hardware and diagnostic integration, and many errors were corrected before attempting to integrate the programs in the system labs.

The software integration plan for the system laboratory recognized that an incremental approach was needed to add and test program modules in an orderly manner, moving from the basic to the complex until all functions were operational.

The first objective in the integration plan was to integrate the 1A Processor utility system (described in 4.5) with the processor to provide

for loading program stores (and disk backup). This was also necessary to obtain memory or register snapshots and program traces on various program or hardware match conditions, and to perform other utility functions.

The next objective was to verify or debug basic system configuration, initialization, and executive control cycling. Next, basic input-output message processing and IO terminal handling was verified. At this point the basic operating system was cycling.

Next, basic capabilities for fault recovery, subsystem configuration, disk administration, and diagnostic paging were verified, followed by verification of the operation of the basic diagnostic tests. With few exceptions, these diagnostic tests became operational very quickly because they had been previously debugged/exercised in ITS, frame testing, and logic simulation. At this point, a preliminary issue of the processor software was released to the No. 4 ESS and No. 1A ESS groups. This allowed each respective application to proceed with its system integration in parallel with the integration of remaining processor functions (such as memory audits, tape administration, error analysis, and field utilities).

4.3 System tests

Most of the processor software integration was done in a system laboratory containing only a processor and system lab utilities. To be sure that the processor would operate properly concurrently with other system functions, a set of processor tests was designed and executed in the No. 4 ESS and No. 1A ESS system labs. For these tests, the system lab was configured to simulate the environment of an operating in-service office. In this environment, the system lab utilities were turned off, all system units were configured on-line with no error detectors inhibited and a simulated call load was turned on.

An exhaustive system test would have exercised all possible system interactions and all possible function overlaps, but, of course, this was not practical. Instead, a set of tests were carefully designed to exercise (with a reasonable amount of system lab time) a wide range of interactions and overlaps. The tests were highly sequential, since early tests had to pass in order to enter later tests with the proper initial conditions. Each test required that a specific system stimulus or sequence of stimuli be applied and that proper system reactions be verified by the tester. Many tests required only input messages from a terminal as a stimulus. Other tests required frame power switch operation, a specific terminal on a frame to be grounded (to stimulate a stuck-at-0 fault), or other special action as a test stimulus.

4.4 Special evaluation and stress tests

In addition to the system tests that were aimed at design verification in a normal operating office environment, special evaluation and stress tests were designed and executed to test certain functions under abnormal or worst-case conditions.

To evaluate the overall fault-recovery and diagnostic program response to hardware faults, special fault-insertion experiments were performed. (These were performed in addition to the physical fault insertion for the trouble-location data-base generation described in Ref. 13). Before the first office cutover, over 2000 randomly selected faults were inserted one at a time in the processor and the resulting fault-recovery and diagnostic responses were evaluated. Many of the faults were reinserted under different initial conditions. For example, faults were inserted in a call store under the following five initial configuration conditions:

(*i*) Active store faulted with a spare store in service and duplicating the faulted store.
(*ii*) Spare call store faulted.
(*iii*) Active call store faulted with a spare store in service but not duplicating the faulted store.
(*iv*) Active call store faulted with all spare stores marked out-of-service.
(*v*) Active call store faulted with a call store bus marked out-of-service.

As a result of these fault-insertion experiments, several problems were uncovered in the fault-recovery and diagnostic programs.

To evaluate the file-store system under worst-case load conditions, a special file-store exercise program was written to carry out experiments in the system lab. Using this program, file-store job requests of different profiles were submitted to the file-store administration program at a high rate over a considerable time interval, which thus simulated the high activity expected under peak call load conditions in No. 4 ESS and No. 1A ESS. These experiments were performed on different system configurations, including cases where a duplicate file-store controller was out of service and all jobs were required to be processed by the remaining controller. The file-store exercise experiments uncovered several subtle software and hardware design problems that were not apparent during integration and system tests because the problems tended to occur only under infrequent, high-activity-related conditions. A tape-exercise program was written to perform similar tests on the tape system and some experiments were conducted with the file-store and tape-exercise programs running simultaneously.

4.5 System laboratories

Two system laboratories were provided for the early stages of 1A Processor integration and testing. One lab was used for initial hardware and diagnostic software integration and, subsequently, as a test bed for verifying hardware change packages. The second lab was used for 1A utility system and 1A Processor software integration. Both labs contained a stand-alone processor (no network frames) with sufficient stores to hold the 1A Processor software.

Later in the project, most of the 1A Processor work was consolidated into a single lab, which is now used for field support, new feature development, and verification of design changes for cost reduction.

4.6 1A Processor utility system

A primary function of a system laboratory is to provide the facilities and the administrative environment in which system programs, written by a large number of programmers, can be tested, debugged, and integrated into a complete software system. The facilities must include a working processor, and a complement of peripheral switching equipment sufficient to allow the exercising of all program functions. In addition, special equipment must be provided to load or modify programs, establish a precise test condition, trigger execution of a program or function to be tested, accurately record resulting program actions, and provide printouts or displays to enable programmers to locate and resolve program bugs. This control and monitoring facility in a system laboratory is referred to as the utility system (which should not be confused with the field utility features provided in the 1A Processor software package).

The 1A Processor utility system (Fig. 7) is comprised of a utility computer (UC) and a utility test console (UTC). The utility system software resides primarily in the utility computer. This software is supplemented with a few special programs resident only in the system laboratory processor. All necessary communications with the 1A Processor can be handled through the UTC, which is a highly complex man-machine and computer-machine interface. Once a system lab becomes fully operational, of course, processor communications typical of a working office (TTY, tape units, master control center, etc.) are also available. The UTC contains data formatting and buffering circuitry through which card input, high-speed printer output and magnetic tape IO from the UC is interconnected with the processor. It also contains complex control and display panels through which either of the central controls (CCs) can be manually controlled and monitored. In addition, the UTC contains a high-speed semiconductor store and a wide variety

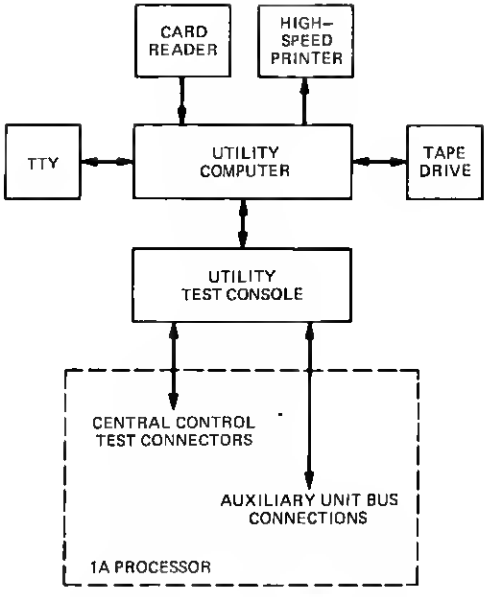

Fig. 7—Flow diagram of 1A Processor utility system.

of matcher circuits by which it autonomously monitors the operation of the processor as it executes a program.

Some discussion of UTC initial design considerations will be useful in explaining its operational features. In general, the reaction of the processor to program instructions may be determined by looking at the contents of the various CC registers after each program step is executed. This could be done by program interrupts that cause the reading of desired registers by a utility program, by slowing or stopping the CC clock for a hardware or a manual readout, or by designing the utility system hardware to be fast enough to do the readout in real time.

A lesson learned in previous ESS systems is that programs do not run the same when they are disturbed by program interrupts. Furthermore, the hardware may not behave the same at nonstandard CC clock rates as it does at standard because relative timing shifts may occur. The utility system had to be designed, then, to be able to monitor processor operations in real time at full system speed.

For a manually controlled operation, a "suspend CC" capability is provided whereby a signal from the UTC halts internal CC sequencers without altering register contents. The release of this signal, therefore, causes program execution to resume at normal speed from exactly the same machine state.

The use of integrated circuits in the 1A Processor has had far-reaching UTC design consequences. A large number of the internal CC registers, which must be monitored, appear only on buried circuit nodes on an integrated-circuit chip; the point does not appear on the backplane where a monitoring lead might be connected. This problem is handled by replication of all such CC registers within the UTC. The data and gating signals for these registers are accessible from the backplane, where they appear briefly on internal CC buses, as the registers are written or read. These and other backplane points are wired to test connectors for UTC access. Hence, once data is read into a CC register, its corresponding follower register in the UTC contains an exact copy of the data; this follower register is connected for continuous monitoring within the UTC.

Since the effect of specific program steps on the contents of specific CC registers must be known to determine the behavior of a program, the UTC is provided with a high-speed semiconductor store, the monitor store (MS), which saves this data. The MS holds 512 words; each word contains 520 bits, which is sufficient to hold the contents of approximately 21 CC registers. Data and addresses are "snapped" into the MS under control of a wide variety of matcher circuits. Masking capability is provided so that a matcher can look for single bits or bit groupings as well as complete words. Various single addresses, address ranges, clock times, or data words that might arise in program operation are prestored in these matcher circuits. When (or if) a bit combination (bit, bits, word, address, or address range) occurs, which a matcher is set up to detect, the matcher produces a trigger signal that causes a "snap" of CC register contents into the MS.

To monitor the central controls in real time, over 900 transmission-line connections tap into each of the CCs via the test connectors; over two gigabits of CC data flows into the UTC each second. The UTC dynamically selects, on a cycle-by-cycle basis, between the two CCs and in the choice of up to 21 of the CC registers. Both "next N" and "last N" program trace-modes are available. In the former case, the MS starts storing data upon receiving a trigger from a particular matcher; in the latter case, the MS continually stores data until it is stopped by a trigger. This allows the recovery of the recent history of program actions that led up to a particular event.

To access the system program-store and call-store memories, the UTC contains a memory-access unit and connections to the processor auxiliary unit (AU) bus. With this facility, the UTC accesses program store or call store using the direct memory access (DMA) circuit in the central control. This access, under control of the UC, is used for loading programs or overwrites in program store, for initializing call store, and for dumping program- or call-store memory as specified by the programmer.

In a typical batch operation, a programmer (or batch operator) reads

Fig. 8—1A Processor utility system.

into the UC a deck of cards that specifies the matcher setup and data-gathering parameters; small program changes (temporary overwrites) may also be entered on cards. New programs or large program changes are entered into the UC on a magnetic tape. The UC then sets up all matchers in the UTC, assigns MS locations to the CC registers to be monitored, causes the UTC to make any necessary changes in the processor program stores, and starts processor program execution. During the run, the UTC autonomously gathers the desired data into the MS. At the end of the run, the UC reads out the MS contents, and outputs this in prescribed formats on the high-speed printer for analysis by the programmer.

As mentioned previously, completely manual operation from the UTC may also be elected for certain difficult system problems for which such an operation is appropriate. Figure 8 shows a typical utility system arrangement in a system lab.

4.7 System change implementation

4.7.1 Trouble reporting

From the early stages of development, monthly status meetings were held to review progress on all aspects of 1A Processor development. Teams were defined for each subsystem with representatives from the circuit-design, diagnostic-design, physical-design, and data-base administration groups. One member, on a rotating basis, gave the monthly status report for his or her team. This team approach eliminated many interface problems, and the status reports tended to highlight serious problems not solved by the team and, thus, requiring the attention of management. When a serious problem arose, the proper experts were convened as a task force to solve the problem, and upper management monitored all organizations to assure coordination and cooperation.

Concurrent with the assembly of the first 1A Processor, a formal trouble-reporting system was implemented. In this system, all observed troubles including faulty circuit packs, wiring errors, hardware-design errors, software-design errors, and general system misbehaviors were documented and entered in the trouble-report data base. For each unresolved trouble, personnel were assigned responsibility for clearing the trouble. Subsequent progress on implementing a "quick fix," as well as a permanent fix, was entered in the data base. A report indicating the status of each trouble was distributed to management on a regular basis, which insured that troubles did not remain outstanding for an unreasonable period of time. Trouble reports on faulty circuit packs and other components were monitored closely for evidence of weakness in design, manufacturing, testing, or handling. Corrections were implemented where appropriate.

4.7.2 Hardware change administration

Early in the project it was recognized that frame or system debugging progress would be severely hampered if the only way to correct a design error was to produce new artmasters and fabricate new circuit packs or backplanes. Therefore, "quick fix" procedures were developed for manually adding or deleting connections on existing beam-leaded circuit-pack ceramics and on printed-wire and wire-wrapped backplanes. Procedures were also provided for adding and deleting chips and their connections on existing ceramics. These quick-fix procedures were also used to repair failed packs in situations where a new spare was not yet available. Using these procedures, modification or repair of a critical circuit pack or backplane could be implemented in a few hours, compared to the period of several weeks that might be required to produce a new artmaster and fabricate a new circuit pack or backplane.

Of course, the long-term permanent fix for errors uncovered in a frame design required complete design update including data base update, simulation, new artmaster generation, etc. Special software was implemented on tbe computer and interactive graphics terminals to assist in generating data base edits that were logically equivalent to quick-fix changes that already had been installed and verified in the frame-test facility or in a system laboratory.

Design-change activity was monitored closely to insure that critical problems were given priority with respect to allocations of resources and that design fixes were thoroughly verified before being propagated to system laboratories and field installations. Approximately two years before the first cutover of the processor, a hardware-change committee was formed composed of No. 4 ESS and No. 1A ESS system representatives, as well as representatives from the processor design groups. The function of this committee was to evaluate the priority and impact of all proposed hardware changes and schedule approved changes in early offices so that cutover schedules were not jeopardized.

4.7.3 Software change administration

A large software development requires special tools and administrative procedures for introducing changes in the software. The general objectives are to expedite corrections for serious bugs and to provide for smooth, minimum-impact introduction of all necessary changes, including new features or enhancements. It was very important that changes in the 1A Processor software be coordinated and introduced in such a way that progress was supported, not hindered, on the No. 4 ESS and No. 1A ESS application system developments.

An important administrative task was to evaluate on a regular basis all outstanding software problems (documented on trouble/failure re-

ports described previously), to determine the seriousness of each problem and allocate resources, to monitor progress on fixes, and to schedule the necessary changes. Similarly, proposed changes for new features or enhancements were reviewed and approved changes were closely monitored and coordinated.

A change to fix a program bug was first tested and then implemented using the program overwrite facility of the 1A utility system. At a later time, the program source was edited and the program was reassembled for final implementation of the change. Program overwrites were installed on a daily basis and program reassemblies and reloads were done on intervals of approximately two months.

Program overwrites were prepared using the new "source overwrite" technique developed for 1A Processor and No. 4 ESS software. With this technique, the programmer prepared new high-level-language program statements to be inserted in the program or to replace old incorrect statements. These statements, along with appropriate insert, delete, and replace commands, were operated on by the source overwrite assembler. The assembler performed a partial program assembly (for the statements being changed), and produced both a program-overwrite deck in 1A machine language as well as a corresponding source-edit deck. The program source data set was not changed until the source-edit deck was applied, and this was not done until the program-overwrite deck was installed and verified in the system lab. The source-overwrite technique was a substantial improvement over previous program-overwrite techniques, since a programmer only had to fix the bug once using the high-level language. Also, new program loads were brought up much more quickly because new assemblies more closely matched the overwritten old assemblies.

Many of the changes to provide new features or enhancements were also first introduced and tested via program overwrites before including the changes in new assemblies and loads. This was not practical for some of the larger changes, which required the assembly of new modules. For these cases, the "binary load" facility of the 1A utility system was used to temporarily place the object module in program store for test purposes.

V. CONCLUSION

The 1A Processor is operating very satisfactorily in its first application in the No. 4 ESS office at Chicago, which was cut over on January 16, 1976. Experience in this office indicates that the design is sound and that objectives for ultrareliable system operation are being achieved. Much of the success of this project is credited to the extensive design verification and testing done at each level of the system's integration.

VI. ACKNOWLEDGMENTS

The authors are reporting on the work of many people in Bell Laboratories and Western Electric. They wish to acknowledge the efforts of all those who developed the extensive computer aids and test facilities as well as those who planned and carried out the testing and integration tasks.

REFERENCES

1. R. E. Staehler, "1A Processor: Organization and Objectives," B.S.T.J., this issue, pp. 119–134.
2. G. Haugk and S. H. Tsiang, "System Testing of the No. 1 Electronic Switching System," B.S.T.J., *43* (September 1974), p. 2575.
3. D. R. Barney, P. K. Giloth, and H. G. Kienzle, "No. 1 ESS ADF: System Testing and Early Field Experience," B.S.T.J., *49* (December 1970), p. 2975.
4. B. P. Donohue, III, and J. F. McDonald, "SAFEGUARD Data-Processing System: Process-System Testing and the System Exerciser," B.S.T.J., 1975, SAFEGUARD Supplement, p. S111.
5. A. E. Spencer, Jr. and H. E. Vaughan, "No. 4 ESS; a Full Fledged Toll Switching Node," International Switching Symposium, Kyoto, Japan, October 1976.
6. J. W. Nowak, "No. 1A ESS—A New High-Capacity Switching System," International Switching Symposium, Kyoto, Japan, October 1976.
7. J. O. Becker et al., "1A Processor: Technology and Physical Design," this issue, pp. 207–236.
8. H. W. Chang, G. W. Smith, and R. B. Walford, "LAMP: System Description," B.S.T.J., *53* (October 1974), p. 1431.
9. S. G. Chappel, "LAMP: Automatic Test Generation for Asynchronous Digital Circuits," B.S.T.J., *53* (October 1974), p. 1477.
10. A. H. Budlong et al., "1A Processor: Control System," B.S.T.J., this issue, pp. 135–179.
11. C. F. Ault et al., "1A Processor: Memory System," B.S.T.J., this issue, pp. 181–205.
12. G. F. Clement et al., "1A Processor: Control, Administrative and Utility Software," B.S.T.J., this issue, pp. 237–254.
13. P. W. Bowman et al., "1A Processor: Maintenance Software," B.S.T.J., this issue, pp. 255–287.